LibreOffice WASM – an Update

A status report from the journey to get LibreOffice into the browser, fully*

WebAssembly

FOSDEM virtual conference, 2021-02-06

Who’s talking?

- Jan-Marek Glogowski - glogow@fbihome.de
- Thorsten Behrens – thorsten.behrens@allotropia.de

Challenges

Challenges

- Size of the resulting WASM binary

  Currently: optimised = 150M, debug = 200M + ~1G separate DWARF info

- Size of the filesystem image

  ~ 100M with all LO fonts, can be stored locally and split if needed

  → less downloads on updates

- WASM thread pool size is four, but Writer is mostly single-threaded
  - Multi-threaded document loading and UNO IPC are disabled
- Heap size tops out at 4GB, because of 32bit pointers
- Development / Debugging environment still very “fresh”

  → mostly work with a normal build; most problems aren’t WASM-specific

Current WASM status

- https://wiki.documentfoundation.org/Development/WASM
  - boils down to: use master, read static/README.wasm.md
- Master can now build and run:
  - A static oosplash + soffice.bin
    System libraries are still linked dynamically.
  - A WASM soffice.(html|wasm)*
  - A WASM vcldemo.(html|wasm) – works mostly

* There are still many bugs to fix / TODOs

Fixed major problems

- gbuild: linking static executables → done
- LO: code dependency loops → done
- LO: static UNO components from build → done
- LO: building a virtual FS image (fonts, UI, config) → done
- Emscripten:
  - Debug build now links in ~30s; not great but manageable
  - Optimized build still needs huge amounts of memory and time, but saves 25% binary size with -O2
  - Always separate debug data, downloaded on demand


allotropia
Open problems

Problems still to tackle

allotropia

- No nested main loops / no blocking of the browser
  - You can run the main loop in a web worker, but then need a separate frontend.
  - Convert dialogs to async + no more Reschedule() calls.

→ SAL_USE_SYSTEM_LOOP=1 make debugrun

→ Easy start: grep `Application::CreateMessageDialog`

- Like commit 972aa39fb976e30ce73065b1eba69f4c78c17855

→ Easy hack to reference: tdf#146919

Problems still to tackle

- Fix the WASM Qt backend:
  - Use qtbase branch 5.15.2+wasm from allotropia Github
  - Much more bugs then estimated
- Alternatives:
  - Port Gtk to WASM

    but you need some kind of compositor for multiple windows..
  - Use the same frontend as COOL… somehow
  - Implement some “WASM-native” VCL plugin

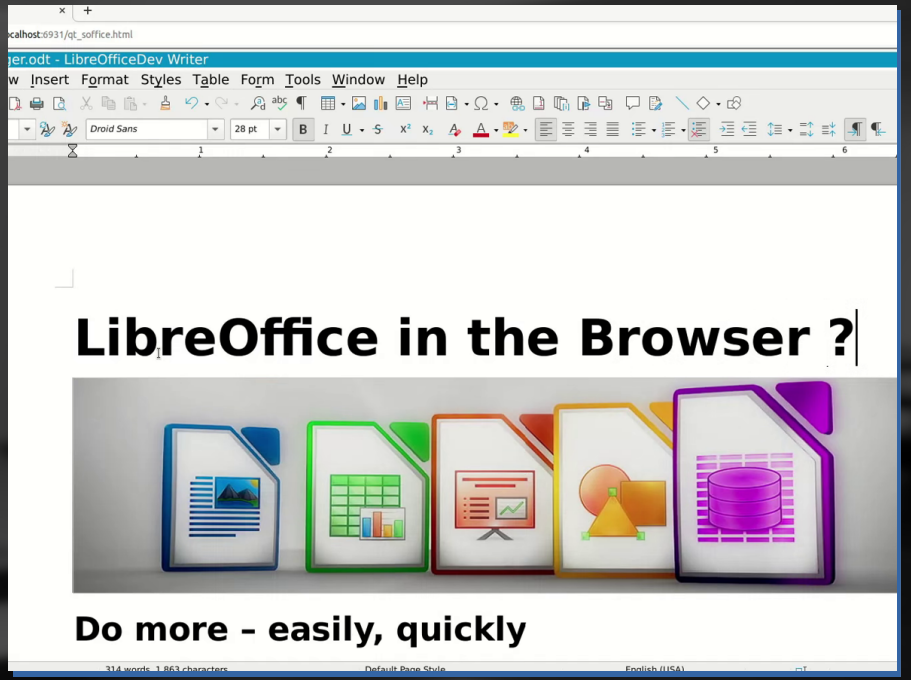
More problems to tackle

allotropia

- “Upload and Download” of local files
- Use browser APIs where / if possible
- Persistent storage for the FS image and user files / data
- Use / Download translations + dictionaries
- Implement a real UNO bridge, probably using WAT
- p2p document editing (as originally planned…)
- Port to a WASI
- Moonshots:
  - Switch to WASM modules AKA “dynloading”
  - Replace gbuild with Meson

LOWA
LibreOffice WebAssembly

- native port of LibreOffice, running client-side in the the browser
- project is funded by NLnet / Horizon 2020, and allotropia




allotropia
Demo!

Updated project plan

allotropia

- switch focus onto JavaScript side:
  - GUI & embedding
  - use browser APIs whereever possible
- get a demo End2End encrypted editing session going by summer 2022
- usable HTML widget for rich text editing by Q3

What to expect
(and our vision for LOWA)

- ***not*** a replacement for desktop/mobile LibreOffice, or Collabora Online
- instead serving unmet needs:
  - your platform is the browser? here's your everything-works text widget!
  - require privacy-by-default, or end2end encryption? here's your no-data-ever-goes-to-any-server solution!
  - want planetary-scale for your product, but lack GAFAM's number of data centers? here's something that scales like a static website!
- want to play yourself? have a look, demo setup here:

https://lab.allotropia.de/wasm


allotropia
nlnet
Questions & Answers